dge

\\server\name

PSCRIPT Page Separator

Responses to Office Action for Application 09/682,315

Inventor: David Ge
Email address: dge@microsoft.com
Phone: (425)707-4224

Date: November 28, 2004

Review by: Ariel Bentolila, Bay Area IP, LLC (Principal member)

Response Summary

My invention must not be obvious.

1. It is shown that codeless programming is much desired in software engineering and is a very difficult research topic. My codeless programming invention has clear advantages over prior-arts, including US Patent 5,850,548 and US Patent 5,862,379, and thus my invention could not possibly be obvious to persons with average skills.
2. All arguments provided in this document are consistent and not contradicting to each other; cross references are also provided when presenting the arguments; thus it is proved that all citations from what taught by Patel, Budd, Linden, and Deitel putting all together cannot provide motivations, hints and suggestions leading to what my invention has achieved and claimed: codeless programming without compiling; and thus my invention could not possibly be obvious to persons with average skills.

Reply Note 1

Regarding Page 2 Item 1

Topic: Drawings

Response:

Bay Area IP, LLC will review and clean up the drawings to make them up to USPTO standard.

Reply Note 2

Regarding Page 2 Item 2 and Item 3

Topic: Specification

Response:

The Abstract was rewritten to make it within 150 words.

Reply Note 3

Regarding Page 3 Item 4 and Item 5

Topic: The formatting of the Claims

Response Summary:

The original claims were reformatted according to USPTO standards.

Response Details:

The claims are rearranged in the ways described below. Regarding referencing non-existing steps D, E, and F, these steps actually exist. When USPTO published my application, the line feeds and carriage returns were removed, and indentations were also removed. It made it hard to read and see these steps being listed. That was why the Examiner thought that these steps did not exist. Actually the last page of the full text image of the publication of my application from USPTO shows correct formatting.

Claim 1 was rearranged in the following way:

1) A period was added at the end of this claim;
2) Added words "in a codeless manner without using of a compiler". These words were in the "Background of Invention" section of the application: "This invention achieves code-less programming by building such a developing and executing environment such that…". The meaning of these words was also implied in the claims 1, 2, 3, and 6.

Claim 2 was rearranged in the following way:

1) A period was added at the end of Claim 2.

Claim 3 was rearranged in the following way:

1) A period was added at the end of Claim 3.
2) Added words "so that the action list becomes an event handler for the event and thus codeless programming is achieved" to explain "event-action-list-mapping". Term "Event-action-list-mapping" is used in my invention in the same meaning of term "event handler" in programming with coding.

Claim 4, 5

1) These two claims were canceled. Their contents were moved into claim 6.

Claim 6 was rearranged in the following way:

1) Claim 4 and claim 5 were canceled and their contents were put in the context building text of claim 6;

2) A period was added at the end of claim 6.

Claim 7 was rearranged in the following way:

1) A period was added after this claim.

Claim 8 was rearranged in the following way:

1) A period was added after this claim.

Claim 9 was rearranged in the following way:

1) A period was added after this claim.

Reply Note 4

Regarding Page 3

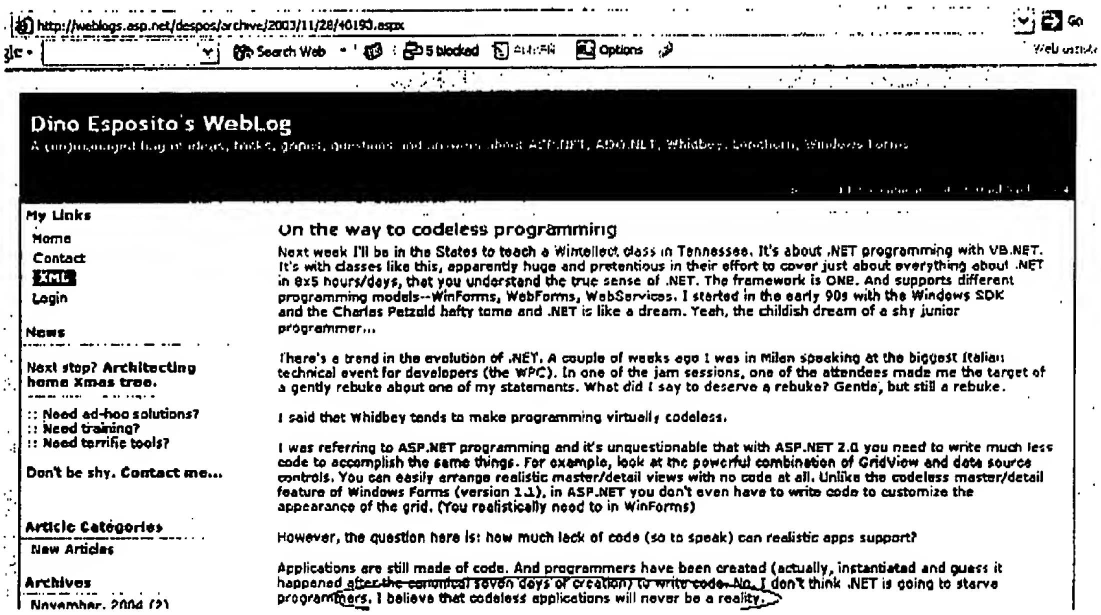
Topic: Claim Rejections

Response Summary:

What taught by Patel, Budd, Linden, and Deitel put all together cannot provide motivations, hints and suggestions leading to what my invention has achieved and claimed: codeless programming without compiling; therefore my invention is not obvious.

Response Details:

1. Patel and Budd did not define the action class and action list class which are precisely defined in my invention; nor did they provide any hints and motivations related to my action class and action list class. My action class and action list class are key components in achieving my codeless programming system which has clear advantages over prior-arts, including taught by Patel and Budd, see Reply Note 6, 7, 8, 9, 10, 12, 13, 14. Therefore my invention is not obvious.
2. Patel and Budd did not use the way described in my invention to do event handling; nor did they provide any hints and motivations related to my way of event handling. My way of event handling is a key component in achieving my codeless programming system which has clear advantages over prior-arts, including taught by Patel and Budd, see Reply Note 10, 15, 16, 17, 18, and 22. Therefore my invention is not obvious.
3. My sorted action list is for execution of actions, one by one, not for action selection. My action class is specifically designed for my way of event handling to achieve codeless programming without compiling. There are no motivations and suggestions in Patel and Budd for creating my action class and action list class; nor did they provide any hints and motivations related to making and executing action class objects and action list class objects. Therefore my invention is not obvious. See Reply Note 7.
4. US Patent 5,850,548 (December 1998) and US Patent 5,862,379 (January 1999) are providing the solutions for the same problem: to create a graphic (preferably codeless) programming system. My invention has clear advantages over these two patents. See **Reply Note 26** below that my invention has clear advantages over US Patent 5,850,548. US Patent 5,862,379 still needs script language while my invention is a complete codeless programming system. "Codeless programming" is still a frontier in software engineering and research. It is not a topic for persons with average skills. Actually in 2003 an expert (Dino Esposito) stated that "I believe that codeless applications will never be a reality." See http://weblogs.asp.net/despos/archive/2003/11/28/40193.aspx.


http://weblogs.asp.net/despos/archive/2003/11/28/40193.aspx

Dino Esposito's WebLog

My Links
Home
Contact
XML
Login

News

Next stop? Architecting home Xmas tree.

:: Need ad-hoc solutions?
:: Need training?
:: Need terrific tools?

Don't be shy. Contact me...

Article Categories
New Articles

Archives
November, 2004 (2)

On the way to codeless programming

Next week I'll be in the States to teach a Wintellect class in Tennessee. It's about .NET programming with VB.NET. It's with classes like this, apparently huge and pretentious in their effort to cover just about everything about .NET in 8x5 hours/days, that you understand the true sense of .NET. The framework is ONE. And supports different programming models--WinForms, WebForms, WebServices. I started in the early 90s with the Windows SDK and the Charles Petzold hefty tome and .NET is like a dream. Yeah, the childish dream of a shy junior programmer...

There's a trend in the evolution of .NET. A couple of weeks ago I was in Milan speaking at the biggest Italian technical event for developers (the WPC). In one of the jam sessions, one of the attendees made me the target of a gently rebuke about one of my statements. What did I say to deserve a rebuke? Gentle, but still a rebuke.

I said that Whidbey tends to make programming virtually codeless.

I was referring to ASP.NET programming and it's unquestionable that with ASP.NET 2.0 you need to write much less code to accomplish the same things. For example, look at the powerful combination of GridView and data source controls. You can easily arrange realistic master/detail views with no code at all. Unlike the codeless master/detail feature of Windows Forms (version 1.1), in ASP.NET you don't even have to write code to customize the appearance of the grid. (You realistically need to in WinForms)

However, the question here is: how much lack of code (so to speak) can realistic apps support?

Applications are still made of code. And programmers have been created (actually, instantiated and guess it happened after the canonical seven days of creation) to write code. No, I don't think .NET is going to starve programmers. I believe that codeless applications will never be a reality.

Clearly codeless programming is much desired in software engineering and is very difficult to achieve. My codeless programming invention has clear advantages over prior-arts and thus my invention could not possibly be obvious to persons with average skills. The software development system using my invention has received much appreciation from users. For example, Albert Mullagulov from Turkmenistan said: "I've just discovered the product you are marketing and, to be honest, I'm amazed at the features of Limnor Application." "Since my programming knowledge (even in VB) is quite restricted, I've been looking for ready-to-use sample codes to design good applications for my business use and when I happened on your application, I realized that it was the very thing I'd sought for.". See http://www.limnor.com/ (go to FAQ) (It is rated 5-stars by several web sites). This product has become the major codeless programming system in the world (searching "codeless programming" in www.google.com, and several search results in the first page will be for this product).

The above points regarding the references will be detailed when responding to each reference below.

Reply Note 5

Regarding Page 4

Topic: Building context for presenting invention in claim 1. Patel pages 265-268, Fig 15.1, Linden page 126.

Response Summary:

Most texts quoted from claim 1 on this page are for building a context to present my invention.

Response Details:

Most texts quoted from claim 1 on this page are for building a context to present my invention. My invention is to define an action class and an action list class as event handler to form a new codeless programming system, which is not obvious for a person with average skills.

Patel pages 265-268 and the figures describe a graphic programming process with Java beans. This

Java bean process cannot be linked to my invention. The following points proved that the approach used by my invention is not obvious:

1. In Patel's process an event of one Java bean is linked directly to a method of another Java bean; the method of the Java bean acts as the event handler. In my invention, an event is linked to an object of action list class (see Claim 3); the members of the action list class are defined by the user interactively (see Claim 1, step A). There is no motivation and suggestion in Patel's teaching to create an action list object as an event handler. Java bean process is not a true codeless process; the development system creates the event linking code; compiling is needed. My process does not involve coding at all, and no compiling is needed.
2. In Patel's process when an event is linked to a method of a Java bean, the development system automatically generates a new adapter class code. Such an adapter class code is unrelated to the action class defined in my invention. My action class is not dynamically created code, and thus no re-compiling is needed for my system to work. My action class can be used for any event handling while Java bean needs different adapter classes for different types of events. Therefore what Patel taught cannot be used or modified to achieve what my invention achieved: codeless programming without compiling.
3. Linden page 126 mentions "every thing is an object". In my approach, an object must have a method to process an action class of my definition, see claim 6, step H3C4: "The said located object class instance carries out the said action method". This is a special requirement only needed for my invention. There are no suggestions and motivations in Linden page 126 to require the generic object to be able to process an action object.

Reply Note 6

Regarding Page 5 – Paragraph 2

Regarding Budd pages 89 – 92

Topic: Action class

Response Summary:

My action and action list classes have deep, wide and fundamental differences from Budd pages 89 – 92. Because of the differences are so fundamental that what taught by Budd cannot be used or modified, directly or indirectly, for the purpose of my invention, and there are not motivations and suggestions from what taught by Budd that can be linked to my invention. Therefore my invention is not obvious.

Response Details:

What currently taught in all literatures including taught by Budd cannot be used or modified for the purpose of my invention, no matter directly, indirectly or as a hint, as shown below:

1. actionListener is an interface. My Action class is a class, not an interface, see Claim 1: "defining an action class and an action list class; the action class has, as its members, action performer, action method, and action data; the action list class contains a sorted list of action

class instances; the action performer is one of the pre-developed object class; the action method is one of the methods supported by the action performer; the action data are the parameters needed by the action method." My action class is a key component in my invention to accomplish codeless programming. It is used to define application behavior and used in handling events (action class itself does not involve events). In Java and Java bean programming, because interface and wrapper class are event type specific it is not possible to modify them to handle all type of events like my action class can; and because interface and wrapper class are used, there is no motivation of come to my idea of action class. In US Patent **5,862,379**, a linking object plus script language are used to achieve codeless (partially codeless because script language is used to define application behavior) graphic programming. Using my action class the application behavior can be controlled without using script language; thus my invention of action class is not obvious; see Reply Note 15. In US Patent ***5,850,548***, property change is used as event linking message to try to make graphical programming system. Limiting program flow behavior to property changes is clearly too restrictive comparing using my action class to define program flow behavior when my action class is forming action list class and used in event handling. See Reply Note 26 for details. This advantage of my invention again proves that my invention of action class is not obvious.

2. In Java if a class implements actionListener it must write code for actionPerformed method, and thus recompiling is needed. In my Action class there is not actual execution code at all. See Claim 1: "the action method is one of the methods supported by the action performer". That is, the actual execution code is defined outside of my action class. Because no coding is needed when defining event handler, this arrangement makes codeless and no-compilation programming possible.
3. In Java if a class implements actionListener then such a class can be passed to a Button object as the Click event handler. But it cannot be used for other kinds of events which expect other interfaces because the Java compiler must match the specific interface. My action class and action list class can be used to handle <u>any</u> events, because in my invention an event is linked not to a function, and thus no need to match event handling interface; this arrangement makes codeless and no-compilation programming possible. See Claim 3: "linking, in response to input from the user, action list instances created in step D to events of the instances of the object classes to form an event-action-list mapping".
4. Because all classes implementing actionListener must be pre-hard-coded, the available actions are limited if used in a codeless graphical manner. My action class does not have execution code itself and it can use <u>any</u> methods supported by all action performer classes which are classes supporting properties-methods-events model; this arrangement makes codeless and no-compilation programming possible.
5. For my invention to work my action class requires an object supporting properties-methods-event model as the action performer, see Claim 1. There is not such a requirement for classes implementing actionListener.
6. In ordinary technologies, executing an event action is simply calling a function, for example, actionPerformed function. In my invention, executing an action can only be done in the

manner defined in Claim 6: "H3c1. Locating the object class instance which is assigned as the action performer for the action; H3c2. Signaling to the said action performer which action method is specified for the action; H3c3. If there are method data specified for the said method of the said located object class instance, the method data are passed to the said object class instance as well; H3C4. The said located object class instance carries out the said action method". This approach does not appear in any literature but it plays a center role in my invention of codeless programming, and thus it cannot be obvious.

7. In ordinary technologies, event parameters, for example, mouse position for a mouse event, are passed into event handler as function parameters directly. In my invention, because event handler is not a function, because event handler is an action list object, event parameters cannot be passed to action list object directly. Claim 7 is my way of using event parameters.

Reply Note 7

Regarding Page 5 – Last paragraph, Page 6 – First paragraph

Regarding Budd pages 227 – 231

Topic: action list class vs menu items

Response Summary:

My action and action list classes are totally unrelated to menu and menu items in Budd pages 227 – 231; my action list class has very narrow and specific definition for executing a set of actions in a specific order, which has nothing to do with menus; therefore my invention is not obvious. What described in Budd pages 227 – 231 cannot be used for building an execution list for a codeless programming system, and thus my invention is not obvious.

Response Details:

Budd is teaching menu items. My action list class has nothing to do with menu items. Menus cannot be used or modified to act as my action list class, and thus my invention is not obvious; as shown below.

1. Menu items are presented to the user to select only one item to execute at runtime. My action list is not a selection list. My action list is an execution list. All actions in the action list will be executed, one by one in a specified order. In this way more than one action can be executed in response to one event. See Claim 6: "sequentially performing each action in the said action list mapped to the said event;". Menu items cannot be used as my action list. So it is clear that menu items are unrelated to my action list.
2. For each menu item a pre-coded-function must be made. In my action list there is not actual execution code. See Claim 1: "the action method is one of the methods supported by the action performer". That is, the actual execution code is defined outside of my action class. That is, menu items have nothing to do with my action list.
3. The ordering of my action list is not for quicker access. It is not for users to visually locate selection, nor to locate an item in the list quickly. I am defining the execution order by specifying which action will be performed first, which action will be performed next, and so on. Execution order is very important in programming. All actions in a list will be performed one by one. See Claim 6. In codeless programming research field no one handles an event